ARTICLE IN PRESS

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Computers & Operations Research ▮▮▮ (▮▮▮▮) ▮▮▮–▮▮▮

computers & operations research

www.elsevier.com/locate/cor

Job scheduling methods for reducing waiting time variance

Nong Ye*, Xueping Li, Toni Farley, Xiaoyun Xu

Information and Systems Assurance Laboratory, Arizona State University, P.O. Box 875906, Tempe, AZ 85287-5906, USA

Abstract

Minimizing Waiting Time Variance (WTV) is a job scheduling problem where we schedule a batch of n jobs, for servicing on a single resource, in such a way that the variance of their waiting times is minimized. Minimizing WTV is a well known scheduling problem, important in providing Quality of Service (QoS) in many industries. Minimizing the variance of job waiting times on computer networks can lead to stable and predictable network performance. Since the WTV minimization problem is NP-hard, we develop two heuristic job scheduling methods, called Balanced Spiral and Verified Spiral, which incorporate certain proven properties of optimal job sequences for this problem. We test and compare our methods with four other job scheduling methods on both small and large size problem instances. Performance results show that Verified Spiral gives the best performance for the scheduling methods and problems tested in this study. Balanced Spiral produces comparable results, but at less computational cost. During our investigation we discovered a consistent pattern in the plot of WTV over mean of all possible sequences for a set of jobs, which can be used to evaluate the sacrifice of mean waiting time while pursuing WTV minimization.

Keywords: Waiting time variance; Job scheduling; Quality of Service (QoS); Computer networks

1. Introduction

This paper supports an overall effort to achieve end-to-end QoS assurance for individual high-priority jobs on an information infrastructure (such as the Internet). We work on QoS models at the local, regional, and global levels. A local level QoS model aims to provide service stability and dependability on individual resources, leading to standard parts, which enable predictable performance. This simplifies QoS assurance at the regional and global levels. This paper represents our local level work and considers a single resource on a computer and network system.

Resources provide services to processes, which are generated in response to users' service requests. In a computer and network system, a router is a resource that provides the service of routing and forwarding packets. Timeliness is an important aspect of QoS on computer networks [1]. Various measures of timeliness exist for services on computer networks, including response time, delay, etc. [2]. Delay usually measures the time that a job spends on a computer or network resource. This can be considered completion time or the time elapsed between the job arriving at and leaving the resource after being serviced. Response time usually measures the time elapsed between sending a service request and receiving a response. For example, in a web browsing application, the response time is the elapsed time between sending a web request and receiving the requested web page. Thus, response time includes the round-trip data

* Corresponding author. Tel.: +1 480 965 7812; fax: +1 480 965 8692.
E-mail address: nongye@asu.edu (N. Ye).

0305-0548/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2005.11.015

transmission time between the web client and server, and delays at the client computer, server, and all routers along the client–server path.

Regardless of the measures considered, computer and network resources do not currently provide dependable timeliness of service. For example, a large variance in web request response times makes the service appear unstable and thus unpredictable from a user's perspective. However, if service is dependable or stable over time, a user can establish an expectation on timeliness of service. Service consistency within the user's expectations will lead to user satisfaction and allow the user to plan a service request in advance so as to get service on time. The goal of this study is to pursue dependability and stability in service timeliness by minimizing its variance. We would like to see the same service request served at a consistent level of timeliness with minimum variance. Merten and Muller [3] stated the importance of minimizing response time variance in computer file organization problems. This study focuses on a job scheduling method to minimize WTV to achieve service stability on an individual resource. It is important to have an efficient heuristic method for computer and network resources since they need to handle many concurrent jobs. For example, a network router may process hundreds of thousands of data packets per second.

We develop two job scheduling methods and define sets of small and large WTV problem instances to compare the performance of our methods with existing ones. During our investigation, we discovered that when the means and variances of job waiting times from all possible sequences in a set of jobs are plotted, a consistent pattern emerges. We present these findings before concluding the paper.

2. Job scheduling

Since most computer and network resources may receive multiple service requests (jobs) at a given time, scheduling is required to determine the order in which jobs are serviced. On computers and networks, a large job may be divided into multiple smaller units for processing in parallel or serial. In this study, we consider these smaller units as unique jobs. Different scheduling methods result in different job sequences, leading to different waiting times and WTV of individual jobs.

When scheduling jobs, we can take into account several factors of individual jobs, such as processing time, priority, due date, and so on [4]. This study considers only the factor of processing time. Since jobs arrive dynamically at computer and network resources, we develop Batch Scheduled Admission Control (BSAC), which changes the dynamic system into a static one [5]. Using the BSAC method, dynamically arriving jobs are admitted into a resource in batches. Hence, in this study we investigate the problem of scheduling a batch of jobs to minimize WTV, given the processing times of those jobs.

Our concern is to provide better QoS in computer systems, thus we review job scheduling literature, which has been studied extensively for managing computer and network resources [6–13]. This is also a well known problem in many other fields, such as manufacturing resources in production environments [4]. There exists theoretical and application oriented analysis of deterministic scheduling problems arising in computer and manufacturing environments [14]. Most existing job scheduling methods aim to optimize the worst case or mean performance of various measures such as execution time, throughput, resource utilization, etc. Statistically, given a sample or population, worst case performance corresponds to the minimum or maximum value, and mean performance corresponds to the mean value. These objectives differ from our objective of minimizing the performance variance. Consequently, such methods cannot be directly applied to the job scheduling problem in this study.

There are a number of studies which consider minimizing the variance of job completion time or completion time variance (CTV) [15–31]. A tabu search-based procedure, a dynamic programming algorithm, and a branch and bound algorithm for the CTV problem have been reported [17,18,32]. Several heuristic methods have been developed for the CTV problem [25,27,33]. There are only a few studies that consider job scheduling to minimize WTV, although a job sequence for WTV and a job sequence for CTV are antithetic [3]. For example, the WTV value of a sequence $(1, 2, \ldots, n-1, n)$ is equal to the CTV value of its antithetical one $(n-1, n-2, \ldots, 1, n)$. Two studies report several heuristic methods of job scheduling for WTV problems and compare their performance on small data sets [3,33]. Other studies prove the V-Shape property of the optimal sequence for CTV and WTV [3,16,33]. In a computer and network system, it is desirable to minimize WTV because CTV relies on the variable sizes of jobs, which is out of our control.

CTV and WTV problems are NP-hard [28]. Thus, cost-efficient heuristic methods of job scheduling are necessary for practical use on computers and networks to schedule jobs efficiently and achieve service stability.

2.1. Problem formulation

Although time sharing of a resource is common on computers and networks, the definition of a WTV problem in this study considers only non-preemptive service. Thus, it does not allow a job currently being processed to be interrupted. Given n jobs in a batch all available at time zero, there are $n!$ possible job sequences. We assume that there is no set-up time for each job to be processed by a resource.

We formulate a WTV problem as an Integer Programming problem. The decision variables are s_{ij}'s, for $i = 1, \ldots, n$ and $j = 1, \ldots, n$, representing a job sequence as well as the position of each job in the job sequence. The binary integer, s_{ij}, is 1 if job j is scheduled at position i, and 0 otherwise. There are n positions in the job sequence since there are n jobs. The job to be scheduled and thus processed first is placed at position 1, the job to be scheduled second is placed at position 2, and so on. The processing time of job j is p_j, which is given. The waiting time of the job at position i is w_i.

Objective Function:

$$\text{Minimize} \quad \frac{1}{n-1}\sum_{k=1}^{n}\left(w_k - \frac{1}{n}\sum_{i=1}^{n} w_i\right)^2 \tag{1}$$

Subject to:

$$\sum_{j=1}^{n} s_{ij} = 1, \quad i = 1, \ldots, n; \tag{2}$$

$$\sum_{i=1}^{n} s_{ij} = 1, \quad j = 1, \ldots, n; \tag{3}$$

$$s_{ij} = 0 \text{ or } 1; \quad i, j = 1, \ldots, n; \tag{4}$$

$$w_1 = 0; \tag{5}$$

$$w_i = w_{i-1} + \sum_{j=1}^{n} s_{i-1,j} * p_j, \quad i = 2, \ldots, n. \tag{6}$$

The objective function is to minimize the sample variance of waiting times of n jobs. Eq. (2) describes the constraint that each position is used exactly once. Eq. (3) indicates that each job is assigned to a position exactly once. Eq. (4) gives the integer constraint. The waiting time of the first job to be processed is 0, which is given in Eq. (5). Eq. (6) defines the waiting time of the job at position i ($i \geqslant 2$), which is the waiting time of the job at position $i-1$ plus the processing time of the job at position $i-1$. Thus, w_i is determined recursively.

Since the WTV problem is NP-hard, there are no efficient search algorithms to find the optimal sequence(s). Among four heuristic methods developed by Eilon and Chowdhury (E&C), two methods produced better performance in reducing WTV for a number of small data sets [33]. We select these methods, E&C1.1 and E&C1.2, for further testing on large data sets. Moreover, we develop two new methods, Balanced Spiral (BS) and Verified Spiral[1] (VS), to make further improvements on E&C1.1 and E&C1.2.

In addition to these methods of job scheduling for WTV problems, there exist many job scheduling methods that target different objectives [4–13]. Some common methods include First In First Out (FIFO), Shortest Processing Time (SPT), Weighted Shortest Processing Time (WSPT), Apparent Tardiness Cost (ATC), and Earliest Due Date (EDD) [4,34]. Among these, SPT requires only information of job processing time. Hence, SPT is included in this study. Other methods require additional information, such as job priority (for WSPT) and job due date (for ATC and EDD), which is not considered in this study, therefore we do not include these methods. We include FIFO because it is most commonly used on computers and networks. Although we assume jobs in a given batch have the same arrival time, we generate a random order of jobs in a batch such that FIFO (where jobs arrive dynamically) can be applied. We do not compare BS

[1] A patent has been filed by Arizona State University for the Verified Spiral method under the name of Yelf Spiral.

and VS methods with tabu search and dynamic programming procedures because they are computationally intensive and not practical for computer and network applications.

2.2. Job scheduling methods for WTV problems

In this section, we describe existing job scheduling methods and introduce our methods: Balanced Spiral and Verified Spiral.

2.2.1. FIFO and SPT

A common method of job scheduling for computer and network resources is First-Come-First-Serve (FCFS) or FIFO, where jobs are serviced in the order in which they arrive. Using FIFO, the job with the earliest arrival time is served first. Jobs with earlier arrival times are served before jobs that arrive later.

Using SPT, jobs are processed in ascending order of processing times. It is well known that SPT minimizes the total completion times of a set of jobs. And, since waiting time = completion time − processing time, it follows that SPT produces an optimal job sequence for minimizing the total, and thus mean, of job waiting times.

2.2.2. E&C1.1 and E&C1.2

Given a pool of n jobs to be scheduled into an empty sequence for an n-job WTV problem, based on the V-shape property of the optimal sequence(s) for WTV problems, E&C1.1 produces such a sequence with the following steps [33]:

1. Remove the job with the longest processing time from the job pool, and place it before the tail (end) of the job sequence, and the tail of the job sequence has one more job in it;
2. Remove the job with the longest processing time from the job pool, place it after the head (beginning) of the job sequence, and the head of the job sequence has one more job in it;
3. Repeat Steps 1 and 2 until the job pool is empty.

Hence, this method places jobs in the sequence in a spiral tail-head fashion, from outside in. For $n = 6$, the sequence is {531246}.

Schrage's conjecture states that there exists an optimal sequence in which the longest job is scheduled last, the second longest is first, the third and fourth longest are last-but-one and last-but-two, respectively, in the sequence [15]. E&C1.2 modifies E&C1.1 by incorporating Schrage's conjecture [33]. After the four longest jobs have been placed in this way in the job sequence, E&C1.1 is applied to remaining jobs in the job pool. For $n = 6$, the sequence is {521346}.

E&C1.2 performs better than or as well as E&C1.1 for fifteen WTV problem instances ranging in size from $n = 6$ to 100 [33]. Note that both E&C1.1 and E&C1.2 are computationally very efficient since they require only the sorting of the jobs by their processing times and then produce the final job sequence directly without computation.

2.2.3. Verified Spiral (VS)

A counter-example of an 8-job problem shows that Schrage's conjecture about the placement of the fourth longest job was not valid [25]. Hall and Kubiak prove Schrage's conjecture about the placement of the first three longest jobs for any problem size [26]. That is, there is an optimal sequence for WTV problems in which the longest job is scheduled last, the second longest is first, and the third longest is last-but-one, or a dual optimal sequence in which the longest job is scheduled last, the second longest is last-but-one, and the third longest is first. Two dual optimal sequences for each of a 5 and 6 job WTV problem is shown in Table 1.

In Verified Spiral, we modify E&C 1.1 by first incorporating Schrage's conjecture and Hall and Kubiak's proof about the placement of the first, second and third longest jobs. And then, we modify the spiral placement of remaining jobs by adding a selection between two positions to place the next job, either before the tail or after the head of the job sequence, based on which position produces a smaller variance of waiting times for jobs already in the sequence, as follows.

Table 1
Optimal sequences of a 5 and 6 job problem

Problem no.	Optimal job sequences (showing processing times of jobs at their respective positions)	Job waiting times	Mean of waiting times	Variance of waiting times
1	4, 3, 2, 5, 6	0, 4, 7, 9, 14	6.8	27.7
	5, 2, 3, 4, 6	0, 5, 7, 10, 14	7.2	27.7
2	5, 4, 2, 3, 6, 7	0, 5, 9, 11, 14, 20	9.8333	48.5667
	6, 3, 2, 4, 5, 7	0, 6, 9, 11, 15, 20	10.1667	48.5667

Suppose an arbitrary job set $P = \{p_1, p_2, \ldots, p_n\}$ needs to be scheduled for a single resource. Assume that the jobs are numbered such that $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_n$.

1. To start, first place job p_n in the last position, job p_{n-1} in the last-but-one position, job p_{n-2} in the first position, and job p_1 in the second position. Now the job sequence becomes $\{p_{n-2}, p_1, p_{n-1}, p_n\}$. The job pool has the remaining jobs $\{p_2, \ldots, p_{n-3}\}$.
2. Remove the longest job from the job pool, place the job either exactly before job p_1 or exactly after job p_1 in the job sequence, depending on which position produces a smaller WTV of the job sequence so far.
3. Repeat Step 2 until the job pool is empty.

The VS method requires more computation than E&C1.1 and E&C1.2 due to the verification of waiting time variances for two possible placements in Step 2.

2.2.4. Balanced Spiral (BS)

To reduce the computational cost associated with the VS method, the BS method replaces the verification of WTV during the placement selection (Step 2). In BS, we maintain the balance of the total processing time of jobs in the left (L) and right (R) side of the sequence, while placing a job from the job pool, as follows:

1. To start, first place job p_n in the last position, and job p_{n-1} in the last-but-one position, and then job p_{n-2} in the first position. Let sequence $L = \{p_{n-2}\}$ and sequence $R = \{p_{n-1}\}$. Note that p_n is not included in R. We denote the sum of the processing times of the jobs in L and R as SUM_L and SUM_R, respectively. The job pool has the remaining jobs $\{p_1, p_2, \ldots, p_{n-3}\}$.
2. If $SUM\_L < SUM\_R$, remove the largest job from the job pool, append the job to the last position of L, and update SUM_L; else if $SUM\_L \geqslant SUM\_R$, remove the largest job from the job pool, add the job to the first position of R, and update SUM_R.
3. Repeat Step 2 until the job pool is empty.

We develop the BS method based on an observation of balanced L and R in the optimal sequence of a special case (Appendix A), and the near-balanced L and R that we obtain at each step when placing a job to construct the optimal sequence for some small-size WTV problem instances.

3. Problems for testing

We test our VS and BS methods on both small-size and large-size problems. There are nine small-size problems and four thousand large-size problems.

Table 2 shows the processing times of jobs in each of the nine small-size problems. In Table 2, the jobs for each problem are placed in a random order, which we consider the original arrival order of the jobs for application of FIFO. The processing times of the jobs in problems 1–4 are integers. The processing times of the jobs in problem 5 are generated using a uniform distribution of real values between 1 and 10. The processing times of the jobs in problem 6

Table 2
Nine small-size problems

Problem no.	Job processing times									
1	2	5	3	6	4					
2	5	2	6	7	4	3				
3	7	3	6	4	2	10	8	9	5	
4	5	3	6	2	7	10	8	4	9	11
5	4.67	8.96	9.09	1.91	8.77	4.44	1.13	6.37	2.25	9.63
6	1.12	0.09	0.68	1.84	0.06	5	0.25	3.03	0.15	0.41
7	5.24	6.2	4.77	3.72	6.73	3.91	4.7	2.82	6.1	6.28
8	9	8	25	21	100	7	13	41	5	10
9	8	13	1	5	19	10	2	18	9	16

are generated using an exponential distribution with $\lambda = 5$. The processing times of the jobs in problem 7 are generated using a normal distribution with mean 5 and standard deviation 1. Problems 8 and 9 are from Eilon and Chowdhury [33]. Hence, in addition to the problems with integer processing times, we also include the problems with both non-integer processing times since the processing time of a job on computers and networks is often computed by dividing the service rate into the job size, which in turn often produces non-integer values. We would like to examine how the VS and BS methods work on the problems with realistic values of processing times.

We generate 4000 large-size problem instances using four different probability distributions of job processing times: normal, exponential, uniform and Pareto. There are 1000 problems for each probability distribution, with 100 jobs in each problem. The processing times of individual jobs in a problem are drawn at random from Normal (576, 10 000), Exponential (576), Uniform (64,1088) and Pareto(1.4, 164.57), respectively. We consider the random order of the jobs generated as the arrival order of those jobs for the application of FIFO. The mean value $576((164.57*1.4/1.4-1)=576)$ for Pareto distribution is chosen because the default IP Maximum Datagram Size on computer networks is 576 octets and is now widely implemented in today's Internet [35]. In addition to probability distributions, such as normal, exponential and uniform, which are commonly used in similar studies, the Pareto distribution is investigated in this study because it is reported that sizes of files requested in web applications (common jobs on computer networks) follow a Pareto distribution [36].

4. Computational results and discussions

For the small-size problems, we can find the optimal sequences for each problem using two methods: an operations research (OR) algorithm, such as brand and bound or dynamic programming, or an enumeration of all possible job sequences. We choose to use the enumeration method for several reasons. First, we have some small-size problems with non-integer processing times. Dealing with floating numbers in the OR algorithms is a well-recognized challenge since different rounding methods may lead to different "optimal" solutions, which may be far from the true optimal solution. Second, in addition to finding the optimal job sequences for those small-size problems, we are also interested in plotting the waiting time means versus variances of all possible job sequences so that we can investigate their relationship as discussed in Section 5.

For problems 1–9, we identify the optimal sequence(s) for minimizing WTV by enumerating all possible sequences for each problem. A more efficient enumeration method is to enumerate only V-shaped sequences. There are 2^{n-3} permutations of V-shaped sequences for an n job problem, since we know the placement of the three longest jobs in the optimal sequence(s). The optimal sequences, mean waiting time and WTV for problems 3–9 are shown in Table 3 (rounded to four significant digits using double precision arithmetic), in addition to those for problems 1 and 2 shown in Table 1.

We test the six job scheduling methods described previously on the nine small problems and our large problem set. Each method produces a job sequence for each problem. For a given problem, let V_{opt} denote the minimum WTV from the optimal sequence, and V_s be the WTV from the job sequence produced by job scheduling method s, where s represents FIFO, SPT, E&C1.1, E&C1.2, VS, or BS. Similarly, we let M_{opt} denote the minimum mean produced by

Table 3
Optimal sequences for problems 3–9

Problem	Optimal sequences										Mean	Variance
3	9	6	5	2	3	4	7	8	10		22.2222	180.4444
	8	7	4	3	2	5	6	9	10		21.7778	180.4444
4	10	7	6	3	2	4	5	8	9	11	27.2000	257.0667
	9	8	5	4	2	3	6	7	10	11	26.8000	257.0667
5	9.09	6.37	4.67	2.25	1.13	1.91	4.44	8.77	8.96	9.63	23.2070	187.7415
	8.96	8.77	4.44	1.91	1.13	2.25	4.67	6.37	9.09	9.63	24.3830	187.7415
6	3.03	0.68	0.25	0.15	0.06	0.09	0.41	1.12	1.84	5.00	4.1330	3.7542
	1.84	1.12	0.41	0.09	0.06	0.15	0.25	0.68	3.03	5.00	3.4970	3.7542
7	6.28	5.24	4.77	3.91	2.82	3.72	4.70	6.10	6.20	6.73	21.6770	189.1656
	6.20	6.10	4.70	3.72	2.82	3.91	4.77	5.24	6.28	6.73	22.0630	189.1656
8	41	13	10	7	5	8	9	21	25	100	73.6	1484.7111
	41	13	9	8	5	7	10	21	25	100	73.4	1484.7111
	25	21	10	7	5	8	9	13	41	100	65.6	1484.7111
	25	21	9	8	5	7	10	13	41	100	65.4	1484.7111
9	18	10	8	5	1	2	9	13	16	19	41	540.4444
	16	13	9	2	1	5	8	10	18	19	41	540.4444

SPT, and M_s be the mean from method s. The percentage value of WTV Deviation (WTVD) of V_s from V_{opt} is defined as follows:

$$WTVD_s = \frac{V_s - V_{\text{opt}}}{V_{\text{opt}}} * 100\%. \tag{7}$$

The percentage value of Waiting Time Mean Deviation (WTMD) of M_s from M_{opt} is defined as follows:

$$WTMD_s = \frac{M_s - M_{\text{opt}}}{M_{\text{opt}}} * 100\%. \tag{8}$$

Hence, $WTVD_s$ indicates how close V_s is to V_{opt}. A smaller $WTVD_s$ value indicates better performance for a job scheduling method on that WTV problem. $WTMD_s$ indicates the sacrifice in the mean waiting time when we pursue minimizing the variance of job waiting times. When multiple optimal sequences for the same problem have different means, we choose the one with the smallest mean.

Table 4 shows performance results for six job scheduling methods on our nine small problems. For each problem, *WTVD* from FIFO and SPT are much larger than *WTVD* from the other four methods and VS produces the smallest *WTVD*. In fact, VS produces the optimal sequence for eight of the nine problems. *WTVD* from BS are less than or equal to E&C1.2 for all but one problem. For each of the nine problems, *WTVD* from E&C1.2 is less than that of E&C1.1, which is consistent with previously published results [33]. Overall, VS and BS perform better than the other methods in reducing WTV. The slightly better performance of VS over BS comes with a larger computational cost.

The *WTMD* columns of VS and BS in Table 4 show how much we sacrifice the mean waiting time when reducing the variance. Note that *WTMD* is affected by M_{opt} which varies with problems. Similarly *WTVD* is affected by V_{opt} which also varies. This is not a problem when we compare *WTMD* or *WTVD* among different methods. When we examine the absolute value of *WTMD* or *WTVD*, the effect of M_{opt} and V_{opt} should be considered.

Each of the large-size problems has 100 jobs, which gives 10 000 variables in the IP problem formulated in Section 2. Existing algorithms in Operations Research (OR) software usually cannot handle a problem of this size to find the optimal solutions within a reasonable amount of time, and we have four thousands such problems. We cannot enumerate all possible job sequences to identify the optimal sequences for these problems. Hence, we cannot compute $WTVD_s$ and $WTMD_s$ for the large-size problems. Instead, we compare the performance of each pair of the six methods by

Table 4
Performance results for small-size problems

Problem	FIFO		SPT		E&C1.1		E&C1.2		VS		BS	
	WTVD (%)	WTMD (%)	WTVD (%)	WTMD (%)	WTVD (%)	WTMD (%)	WTVD (%)	WTMD (%)	WTVD (%)	WTMD (%)	WTVD (%)	WTMD (%)
1	48.01	16.67	13.72	0.00	0.36	23.33	0.00	20.00	0.00	13.33	0.00	13.33
2	75.22	38.00	19.15	0.00	0.27	26.00	0.00	22.00	0.00	18.00	0.00	18.00
3	41.70	25.64	31.07	0.00	0.23	32.05	0.03	28.85	0.00	25.64	0.00	25.64
4	32.65	13.81	34.08	0.00	0.19	33.33	0.09	27.62	0.00	27.62	0.00	27.62
5	47.50	58.12	48.07	0.00	0.11	44.99	0.09	35.58	0.00	43.44	0.00	43.44
6	487.41	188.79	66.45	0.00	2.41	139.20	0.01	120.97	0.00	87.51	1.37	95.39
7	13.42	18.23	17.90	0.00	0.02	15.29	0.01	12.26	0.00	14.61	0.00	14.61
8	473.83	135.55	35.31	0.00	1.30	68.31	0.01	56.75	0.00	40.04	0.01	40.90
9	52.44	41.49	51.14	0.00	0.40	53.19	0.17	43.26	0.01	43.97	0.01	43.97

Table 5
Performance results for large-size problems with the normal distribution of job processing times

	SPT	E&C1.1	E&C1.2	VS	BS
FIFO	51.9	100	100	100	100
SPT		100	100	100	100
E&C1.1			98.8	100	100
E&C1.2				100	100
VS					7.9

computing the percentage of the 1000 problems, for a given distribution, in which one method in the pair produces better results (smaller WTV) than the other.

Note that searching for the optimal solutions of an OR problem involving non-integer values such as all of our large-size problems involving non-integer job processing times often presents a challenge since different rounding methods might produce different "optimal" solutions which may differ from the true optimal solutions. Researchers usually employ one of two methods to address this challenge. In the first method, researchers round non-integer values to the closest integer value and then apply a given OR algorithm to the problem with these converted values as input. In the second method, researchers perform an error analysis after applying a given OR algorithm to the problem, using rounding of non-integer values in each step of the computation. This is not an issue in our study, because our purpose is not searching for the optimal solutions of these large-size problems. We only desire to apply the VS and BS methods to these problems to obtain (not necessarily optimal) job sequences for comparing WTV with that of job sequences produced by other heuristic methods. It is important for us to examine the performance of VS and BS job sequences by applying these methods directly with the original, non-integer job processing times to make our results convincing to computer and network engineers, because, in practice, these methods will be applied to realistic, non-integer job processing times on computers and networks.

Specifically, we use a statistical software package, Statistica, to generate real value of the job processing times and we keep them with three decimal digits. We implement the programs of the VS and BS methods in Visual Basic 6. We use "double" as the data type for the job processing times and WTV. In Visual Basic 6, double-precision floating-point data type has 8 bytes as the storage size and ranges from −1.79769313486231E308 to −4.94065645841247E-324 for negative values and 4.94065645841247E-324 to 1.79769313486232E308 for positive values. We use double-precision floating-point data type during the intermediate computation. We keep three decimal digits when reporting the results.

Tables 5–8 show the performance results of each method for the large size problems under each distribution. Each of Tables 5–8 show the percentage of the 1000 problems for a given distribution in which the method in each column produces better results than the method in each row. For example, the last column of Table 5 shows that BS is better than FIFO, SPT, E&C1.1 and E&C1.2 for 100% of the 1000 problems with normal distribution, but is better than VS for only 7.9% of these problems.

Table 6
Performance results for large-size problems with the exponential distribution of job processing times

	SPT	E&C1.1	E&C1.2	VS	BS
FIFO	98.3	100	100	100	100
SPT		100	100	100	100
E&C1.1			100	100	98.6
E&C1.2				100	96.7
VS					1.2

Table 7
Performance results for large-size problems with the uniform distribution of job processing times

	SPT	E&C1.1	E&C1.2	VS	BS
FIFO	25.5	100	100	100	100
SPT		100	100	100	100
E&C1.1			99.1	100	100
E&C1.2				100	100
VS					13.4

Table 8
Performance results for large-size problems with the Pareto distribution of job processing times

	SPT	E&C1.1	E&C1.2	VS	BS
FIFO	100	100	100	100	100
SPT		100	100	100	100
E&C1.1			100	98.1	69.6
E&C1.2				93.7	62.4
VS					0.1

Table 9
Mean and standard deviation of WTVD from the lower bound for large-size problems

	WTVD							
	Normal		Exponential		Uniform		Pareto	
	Mean (%)	STD (%)	Mean (%)	STD (%)	Mean (%)	STD (%)	Mean (%)	STD (%)
FIFO	15.98	2.27	158.30	31.52	57.47	9.68	988.73	10053.53
SPT	15.80	1.50	99.54	7.12	62.55	5.22	48.98	7.88
E&C1.1	0.00	0.00	0.02	0.01	0.00	0.00	0.12	0.81
E&C1.2	0.00	0.00	0.01	0.01	0.00	0.00	0.09	0.76
BS	0.00	0.00	0.00	0.02	0.00	0.00	0.41	2.28
VS	0.00	0.00	0.00	0.00	0.00	0.00	0.08	0.77

Tables 5–8 show that VS and BS are better than FIFO and SPT for all problems (100%) for each of the four distributions. In Table 9, we compare WTV from six scheduling methods with the lower bound on WTV by replacing V_{opt} in Eq. (7) with the lower bound [20]

$$LB = \begin{cases} \dfrac{1}{n-1} \displaystyle\sum_{i=1}^{(n-1)/2} (p_1 + p_2 + \cdots p_{2i})^2/2 & \text{if } n \text{ is odd,} \\ \dfrac{1}{n-1} \displaystyle\sum_{i=1}^{(n-2)/2} (p_1 + p_2 + \cdots p_{2i+1})^2/2 & \text{if } n \text{ is even.} \end{cases} \tag{9}$$

Table 10
Computational time comparison of the scheduling methods (in milliseconds)

Number of jobs	FIFO	SPT	E&C1.1	E&C1.2	BS	VS
10	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
50	0.0000	0.0000	0.0000	0.0000	0.0000	0.6250
100	0.0000	0.0000	0.0000	0.0000	0.0000	1.5625
500	0.0000	0.1563	0.4688	0.3125	0.3125	18.9063
1000	0.0000	0.3125	0.3125	0.6250	0.6250	72.5000
5000	0.1563	2.5000	3.5938	4.3750	2.5000	1489.0630

Note that the lower bound for the WTV problem with n jobs in Eq. (9) is equivalent to the definition of the lower bound for the CTV problem with $n + 1$ jobs [20]. And the CTV and WTV problems have the same optimal variance value [3]. Table 9 shows the mean and standard deviation of WTVD using the lower bound for our large problem set. From Table 9 we see that on average the WTVD from VS are closest to the lower bounds for all four distributions, followed by those from BS (BS is worse than E&C1.1 and E&C1.2 only for the Pareto distribution).

In summary, the results show that for our problems, VS consistently produces the best performance, and overall, VS and BS outperform the other methods, producing solutions equal or close to the optimal solutions.

We give a computational cost comparison of the scheduling methods in Table 10. Our implementation uses Visual Basic on a PC with a 1.9 GHZ CPU and 1 G RAM. We test problems with different numbers of jobs, from 10 to 5000, and run 100 problems for each level using average computational time as the computational cost. As expected, FIFO is the fastest scheduling method as the others need more time for sorting. For these, we use a quick sort algorithm with time complexity $O(n \log n)$. Since VS needs to calculate WTV to verify the positions of jobs, its computational cost is larger than the other methods, which have similar computational costs from sorting. For smaller problems, VS is still a good choice since it gives the best performance with respect to reducing WTV. For applications with rigid computational requirements or a large number of jobs, BS can be applied to reduce WTV with a more acceptable computational cost.

5. Consistent pattern discovery

During our investigation, we discovered a consistent eye-shaped pattern when we plot the waiting time variance over mean of all possible job sequences for a given problem. This pattern allows us to evaluate the mean sacrifice while pursuing minimum WTV, and possibly develop a mathematical method to estimate the minimum WTV and derive the optimal job sequence. We illustrate the pattern and discuss its implications.

5.1. Illustration of pattern

For our problem instances in Table 2, we enumerate all possible job sequences and plot the variance over the mean of the jobs' waiting times. The plots are shown in Fig. 1. We notice that plots (a), (b), (c), (d) and (i) follow a consistent pattern, and that the pattern's appearance thickens as the problem size increases. Plots (e) and (g) also hold the pattern, but not as smoothly and (f) and (h) have a similar shape, but with a sharper curve on the upper contour.

Fig. 2 shows the same pattern for four additional WTV problem instances with increasing differences in the job processing times of a batch. We see that as the difference in processing times increases from plot (a) to plot (d), the distribution of data points in the plot becomes more localized, generating more "gaps" of various sizes in the variance over mean plot. In spite of these gaps, the pattern is consistent. Hence, for all the plots we observe the eye-shape pattern.

5.2. Implication of pattern

One important implication of the eye-shape pattern is that it gives us a new way to evaluate the mean sacrifice while pursuing minimum WTV. For our problems in Table 2, we examine and compare the mean waiting times of three data points in each of the eye-shape plots as shown in Table 10. The first data point is the minimum WTV, located at the

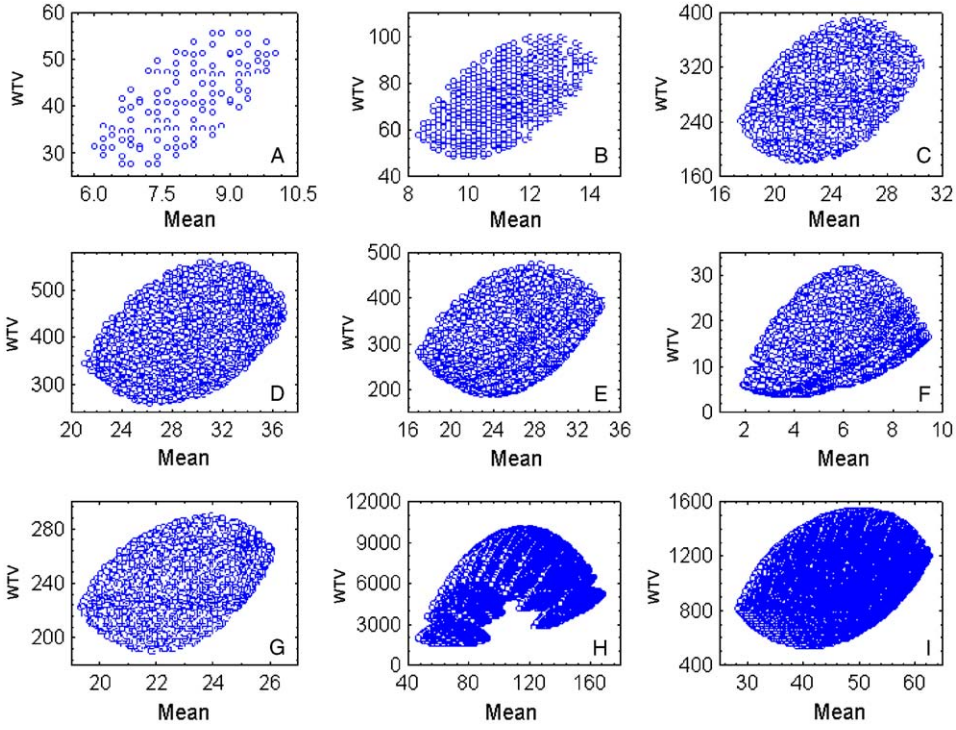

Fig. 1. The variance over mean waiting time plots for problems 1–9.

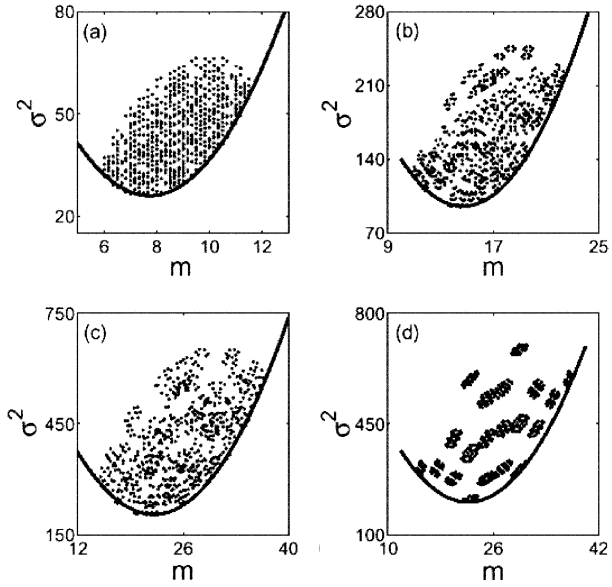

Fig. 2. More examples of the eye-shape: (a) $P = (1, 2, 3, 4, 5, 6)$, (b) $P = (2, 3, 5, 9, 10, 11)$, (c) $P = (1, 3, 9, 14, 16, 19)$, and (d) $P = (2, 3, 4, 17, 18, 19)$.

lowest point on the y-axis (if there are multiple such points, we select the one with the smallest mean value). The second point is the minimum mean, located at the left corner of the pattern, and the third is the maximum mean, located at the right corner of the pattern.

Table 11
Comparison of mean waiting times of three data points

Problem	Data point			Range	Distance of data points 1 and 2	Ratio of distance to range
	1	2	3			
1	6.80	6.00	10.00	4.00	0.80	0.20
2	9.83	8.33	14.17	5.83	1.50	0.26
3	21.78	17.33	30.67	13.33	4.44	0.33
4	26.80	21.00	37.50	16.50	5.80	0.35
5	23.21	17.00	34.50	17.50	6.21	0.35
6	3.50	1.87	9.50	7.64	1.63	0.21
7	21.68	19.25	26.17	6.92	2.43	0.35
8	65.40	46.70	168.40	121.70	18.70	0.15
9	41.00	28.20	62.70	34.50	12.80	0.37

We know that the second data point corresponds to the job sequence produced by SPT and the third corresponds to the job sequence produced by Longest Processing Time (LPT) first [4]. These two points define the range of the mean. From Fig. 1, we observe that the first point is closer to the second than to the third on the x-axis. This is also shown in Table 11, which shows the ratio of this distance to the range. The minimum and maximum ratios are 0.1537 and 0.3710, respectively. The average and variance are 0.2873 and 0.0067.

For these nine problems, we find that on average the job sequence with minimum WTV sacrifices the mean waiting time by about 28.73% of the entire range of means, Hence, while pursing minimum WTV, we do not sacrifice much of the optimal mean.

Two other important implications of the eye shape pattern lie in the lower contour and the use of this contour to find the minimum WTV and derive the job sequence that produces it. If we can mathematically derive the minimum WTV using the lower contour, one implication is that we can compute the minimum WTV without knowing the exact job sequence that produces it. This allows us to evaluate the WTV of a job sequence produced by a heuristic job scheduling method. Further deriving the job sequence that produces the minimum WTV leads to another implication; we will have a mathematical method of deriving the optimal sequence for a given problem, rather than using a space search or heuristic method to find a sequence with good or near optimal WTV.

6. Conclusion

In this study we investigate the WTV minimization problem for a single resource on computers and networks. We develop two methods, VS and BS, which incorporate properties for optimal sequences for WTV problems. By testing and comparing VS and BS with four other job scheduling methods, we demonstrate that VS consistently produces the best performance for the WTV problems tested in this study, followed by BS which has less computational cost, which make it more practical for use on computers and networks due to its comparable performance to that of VS. Although the improvements seem marginal, we anticipate these methods to be useful for application on computer and network resources where the rapidly increasing volumes of data, and speed of computation, justify even nominal improvements to existing methods.

VS and BS have wide applications in managing computer and network resources to provide QoS with regards to service stability and performance dependability for jobs requesting services from those resources. For example, VS and BS can be applied to a router for scheduling data packets to use the transmission bandwidth—the resource of the router, a web server for scheduling services of web requests, and so on. Although we develop VS and BS in the context of Quality of Service on computers and networks, these job scheduling methods for service stability are applicable in many other application fields that demand service stability and performance dependability. For example, VS and BS can be applied to manufacturing production planning for scheduling jobs in a batch on a manufacturing machine.

During our investigation, we discovered that a plot of the variance over mean waiting times of job sequences for a given WTV problem consistently follows a pattern for all the problems in our study. We illustrate this pattern and discuss

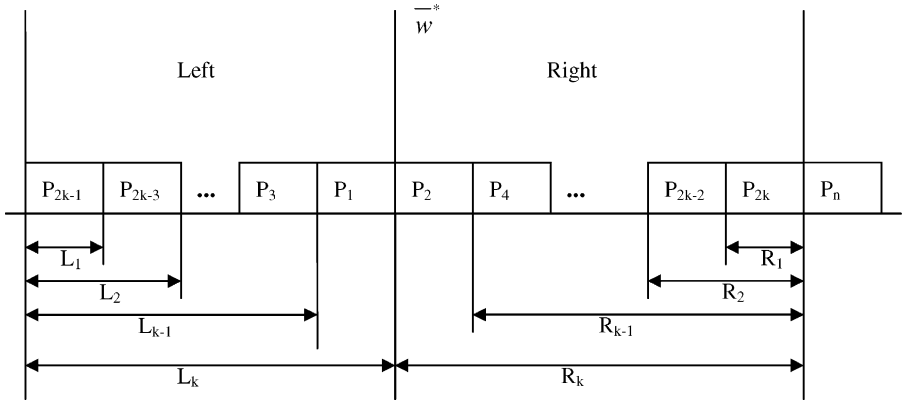

Fig. 3. An optimal job sequence for a special case of WTV problems with $P_{2k-1} = P_{2k}$ and the balance between the left and right part of the optimal schedule.

its implications. We use this observation to show the sacrifice in mean produced by pursuing WTV minimization. While this information may not be difficult to evaluate, our discovery gives another way to look at the problem, and possibly derive a method for determining the optimal sequence.

In this study we consider only a single resource and jobs with only information of their processing times. This paper focuses on the VS and BS algorithms. In an other paper, we consider the relation between the mean waiting time and its variance, as well as the factors contributing to WTV [37]. In our further studies, we extend VS and BS to WTV problems involving multiple resources that jobs need to go through in parallel [38]. We will also investigate WTV as a function of the batch size and the distribution of job processing times so that for a desired value of waiting time variance and known distribution of job processing times we will be able to derive the right batch size for producing a desired value of WTV for a given resource with a known distribution of job processing times. Hence, this study is the first step of our research work towards realizing the goal of service stability and performance dependability on computers and networks as well as systems in many other application fields.

Acknowledgements

This work is sponsored by the Department of Defense (DoD) and the Air Force Office of Scientific Research (AFOSR) under Grant number F49620-01-1-0317. The US government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either express or implied, of, DoD, AFOSR, or the US Government. The authors would like to thank Patrick Hurley and John Faust at the Air Force Research Laboratory for their collaboration with us and their inputs to our research work.

Appendix A

We consider the problem of schedule n independent jobs on a single machine so as to minimize the waiting time variance (WTV). Without loss of generality, we assume that $p_1 \leqslant p_2 \leqslant \cdots \leqslant p_n$. Consider a special WTV problem where n is an odd number and $p_{2k-1} = p_{2k}$, for $1, 2, \ldots, (n-1/2)$. An optimal schedule for this problem is illustrated in Fig. 3.

The job sequence in Fig. 3 is optimal because for this special WTV problem there are only three possible job sequences that satisfy the V-shape property of an optimal sequence [33]: (1) one in Fig. 3, (2) the job sequence of $p_n, p_{n-1}, \ldots, p_2, p_1$, and (3) the job sequence of $p_1, p_2, \ldots, p_{n-1}, p_n$. Hall and Kubiak [26] prove that in an optimal sequence the first longest job is scheduled last, the second longest is first, and the third longest is last-but-one, or a dual optimal sequence in which the first longest job is scheduled last, the second longest is last-but-one, and the third

longest is first. Hence, (2) and (3) are not optimal because they violate the positions of the three longest jobs in an optimal sequence. (1) in Fig. 3 is the optimal job sequence for this special WTV problem.

In Fig. 3, *MS* is the makespan of the optimal job sequence as follows:

$$MS = \sum_{i=1}^{n} p_i$$

and $\overline{w}^*$ is the mean waiting time of the optimal job sequence

$$\overline{w} = \frac{\sum_{i=1}^{n} w_i}{n} = \frac{1}{2}(MS - p_n).$$

Notice that in this job sequence, excluding p_n, we have the rest of the jobs balanced with respect to the middle point as indicated by $\overline{w}^*$ in Fig. 3. That is, the jobs scheduled before $\overline{w}^*$ are completely symmetric to the jobs scheduled after $\overline{w}^*$. Since the optimal job sequence is symmetric, we have the balance between the left part (before $\overline{w}^*$) and the right part (after $\overline{w}^*$) in the job sequence, $L_k = R_k$ for any given pair of L and R, as shown in Fig. 3. This implies that in this special case of WTV problems one way to achieve the optimal job sequence is to maintain the balance between the left part and the right part of the job sequence as in BS. For WTV problems without the property of $p_{2k-1} = p_{2k}$, BS may not produce the optimal job sequence, but may still work to produce a job sequence close to the optimal one as shown in our testing results.

References

[1] Ye N. QoS-centric stateful resource management in information systems. Information Systems Frontiers 2002;4(2):149–60.
[2] Chen Y, Farley T, Ye N. QoS requirements of network applications on the Internet. Information, Knowledge, Systems Management 2003;4(1).
[3] Merten AG, Muller ME. Variance minimization in single machine sequencing problems. Management Science 1972;18:518–28.
[4] Pinedo M. Scheduling theory, algorithms and systems. Englewood Cliffs, NJ: Prentice-Hall; 1995.
[5] Ye N, Bashettihalli H, Li X, Farley T. Batch scheduled admission control for service dependability of computer and network resources. Information, Knowledge, Systems Management, submitted for publication.
[6] Maheswaran M. Quality of service driven resource management algorithms for network computing. International Conference on Parallel and Distributed Processing Technologies and Applications, 1999.
[7] Anderson EJ, Potts CN. On-line scheduling of a single machine to minimize total weighted completion time. Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, 2002.
[8] Rana OF, Winikoff M, Padgham L, Harland J. Applying conflict management strategies in BDI agents for resource management in computational grids. Proceedings of the Australasian Conference on Computer Science, 2002.
[9] Alhusaini AH, Prasanna VK, Raghavendra CS. A unified resource scheduling framework for heterogeneous computing environments. Eighth Heterogeneous Computing Workshop, 1999.
[10] Smith W, Foster I, Taylor V. Scheduling with advanced reservations. International Parallel and Distributed Processing Symposium, 2000.
[11] Hollingsworth J, Maneewongvatana S. Imprecise calendars: an approach to scheduling computational grids. International Conference on Distributed Computing Systems, 1999.
[12] Casavant TL, Kuhl JG. A taxonomy of scheduling in general-purpose distributed computing systems. IEEE Transactions on Software Engineering 1988;14(2):141–54.
[13] Adam TL, Chandy KM, Dickson JR. A comparison of list schedules for parallel processing systems. Communications of the ACM 1974;17(12):685–90.
[14] Ecker KH, Pesch E, Schmidt G, Weglarz J, Blazewicz J. Scheduling computer and manufacturing processes. Berlin: Springer; 1996.
[15] Schrage L. Minimizing the time-in-system variance for a finite jobset. Management Science 1975;21(5):540–3.
[16] Mittenthal J, Raghavachari M, Rana AI. V- and Λ-shaped properties for optimal single machine schedules for a class of non-separable penalty functions. European Journal of Operational Research 1995;86(2):262–9.
[17] Al-Turki U, Fediki C, Andijani A. Tabu search for a class of single-machine scheduling problems. Computers and Operation Research 2001;28(12):1223–30.
[18] Viswanathkumar G, Srinivasan G. A branch and bound algorithm to minimize completion time variance on a single processor. Computers and Operations Research 2003;30(8):1135–50.
[19] De P, Ghosh JB, Wells CE. Scheduling to minimize the coefficient of variation. Internatinal Journal of Production Economics 1996;44:249–53.
[20] Kubiak W, Cheng J, Kovalyov MY. Fast fully polynomial approximation schemes for minimizing completion time variance. European Journal of Operational Research 2002;137(2):303–9.
[21] Kubiak W. New results on the completion time variance minimization. Discrete Applied Mathematics 1995;58(2):157–68.
[22] Cai X. V-shape property for job sequences that minimize the expected completion time variance. European Journal of Operational Research 1996;91(1):118–23.
[23] Manna DK, Prasad VR. Pseudopolynomial algorithms for CTV minimization in single machine scheduling. Computers and Operations Research 1997;24(12):1119–28.

[24] Lu X, Sitters RA, Stougie L. A class of on-line scheduling algorithms to minimize total completion time. Operations Research Letters 2003;31(3):232–6.
[25] Kanet JJ. Minimizing variation of flow time in single machine systems. Management Science 1981;27(12):1453–9.
[26] Hall NG, Kubiak W. Proof of a conjecture of Schrage about the completion time variance problem. Operations Research Letters 1991;10(8): 467–72.
[27] Vani V, Raghavachari M. Deterministic and random single machine sequencing with variance minimization. Operations Research 1987;35(1): 111–20.
[28] Kubiak W. Completion time variance minimization on a single machine is difficult. Operations Research Letters 1993;14(1):49–59.
[29] Manna DK, Prasad VR. Bounds for the position of the smallest job in completion time variance minimization. European Journal of Operational Research 1999;114(2):411–9.
[30] Mosheiov G. Minimizing mean absolute deviation of job completion times from the mean completion time. Naval Research Logistics 2000;47(8):657–68.
[31] Jurisch B, Kubiak W, Jozefowska J. Algorithms for minclique scheduling problems. Discrete Applied Mathematics 1997;72(1–2):115–39.
[32] De P, Ghosh JB, Wells CE. On the minimization of completion time variance with a bicriteria extension. Operations Research 1992;40(6):1148.
[33] Eilon S, Chowdhury IG. Minimizing waiting time variance in the single machine problem. Management Science 1977;23:567–74.
[34] Ye N, Gel ES, Li X, Farley T, Lai YC. Web server QoS models: applying scheduling rules from production planning. Computers and Operations Research 2005;32:1147–64.
[35] Postel J. The TCP maximum segment size and related topics. RFC 879, 1983; URL: http://rfc.net/rfc879.html
[36] Arlitt MF, Williamson CL. Web server workload characterization: the search for invariants. Proceedings of the ACM SIGMET-RICS, 1996.
[37] Li X, Ye N, Xu X. Influencing factors of job waiting time variance for minimizing waiting time variance, submitted for publication.
[38] Xu X, Ye N. Minimization of job waiting time variance on identical parallel machines, submitted for publication.